TUTORIAL 1 (DAA)

Name:- Purnima Kabadwal 1.1

Sec-G Roll-No-~~07~~

Answer 1:

Asymptotic Notation: Asymptotic Notation are the mathematical notions used to describe the running time of an algorithm.

Different types of asymptotic Notations:-

1. Big-O-Notation: It represents upper bound of an algorithm.

$f(n) = O(g(n))$ if $f(n) \leq c * g(n)$

2. Omega notation ($\Omega$): It represents lower bound of Algorithm. $f(n) = \Omega(g(n))$

If $f(n) \geq c * g(n)$

3. Theta Notation ($\Theta$): It represents upper and lower bound of algorithm.

$f(n) = \Theta(g(n))$ If $c_1 g(n) \leq f(n) \leq c_2(g(n))$

Answer 1:

```
for (i=0 to n)
{
    i = i * 2
}
```

i=1
i=2
i=4
i=8
i=16
:
i=n

It is forming GP

$a_n = a r^{n-1}$

$n = a r^{k-1}$

$n = 1 \times (2)^{k-1}$

$\log n = \log 2^{k-1}$

$\log n = (k-1) \log 2$

$\boxed{k = \log n + 1}$ $\quad O(\log n)$

$a_n = n$
$r = 2$
$a = 1$

Answer 8:

$T(n) = 3T(n-1)$

$T(1) = 3T(0)$

$T(1) = 3 \times 1$

$T(2) = 3T(1) = 3 \times 3 \times 1$

$T(3) = 3 \times T(2) = 3 \times 3 \times 3$

$\vdots$

$T(n) = 3 \times 3 \times 3 \cdots$

$3^n$

$= O(3^n)$

$T(0) = 1$

$T(1) = 2T(0) - 1$

$T(1) = 2 - 1 = 1$

$T(2) = 2 - 1 = 1$

$T(3) = 2(T(2)) - 1$

$= 02 - 1 = 1$

$\vdots$

$T(n) = 1$ $\quad O(1)$

Answer 5:



```
int i=1, s=1
while (s<=n)
{
    i++;
    s=s+i;
    printf("#");
}
```

| | |
|---|---|
| i=1 | s=1 |
| i=2 | s=1+2 |
| i=3 | s=1+2+3 |
| i=4 | s=1+2+3+4 |
| i=5 | s=1+2+4+5 |
| i=6 | ⋮ |
| i=7 | ⋮ |
| ⋮ | ⋮ |

Loop ends when $s > n$

$$1+2+3+4 \cdots k > n$$

$$\frac{k(k+1)}{2} > n$$

$$k^2 > n$$

$$k > \sqrt{n}$$

$$O(\sqrt{n})$$

Answer 6:

```
void function(int n)
{ int i, count = 0;
  for(int i=1; i*i<= n; i++)
      count++;
}
```

i=1
i=2
i=3
i=4
⋮
i=k

loop end when

$i * i > n$

$k * k > n$

$k^2 > n$

$O(n) = \sqrt{n}$

Answer 7

```
Void function (int n)
{ int i, j, k, count = 0;
  for (i = n/2 ; i <= n ; i++)
    for (j = 1 ; j <= n ; j = j * 2)
      for (k = 1 ; k <= n ; k = k + 2)
        count++;
}
```

1st loop = $i = \frac{n}{2}$ to $n$ , $i++$

$= O(\frac{n}{2}) = O(n)$

2nd loop : $j = 1$ to $n$ , $j = j * 2$

$j = 2$

$j = 4$

$\vdots$

$j = n$

$O(\log n)$

3rd loop $k = 1$ to $n$ $k = k * 2$

$k = 1$

$k = 2$

$k = 4$ $O(\log n)$

Total Complexity $= O(n \times \log n \times \log n) = O(n \log^2 n)$

Answer - 8



```
function(int n)
{
   if(n==1) return ;  ——— 1
     for(int i=1 to n)
     { for int j=1 to n) ——— n^2
          {
            printf(" ");
          }
     } function (n-3) ——— T(n-3)
}
```

$$\boxed{T(n) = T(n-3) + n^2}$$

$$T(1) = 1$$

$T(1) = 1$

$T(4) = T(4-3) + 4^2$

$= T(1) + 4^2 = 1^2 + 4^2$

$T(7) = T(7-3) + 7^2$

$= 1^2 + 4^2 + 7^2$

$T(10) = T(10-3) + 10^2$

$= 1^2 + 4^2 + 7^2 + 10^2$,

So, $T(n) = 1^2 + 4^2 + 7^2 + 10^2 \cdots n^2 = \frac{n(n+1)(2n+1)}{6}$

$= O(n^3)$

also for terms like $T(2)$, $T(3)$, $T(5)$

$T(n) = O(n^3)$

Answer 9: void function (int n)

```
{
  for(int i=1 to n)
    {
      for (j=1 ; j<=n; j=j+1) —n
        printf ("*").
    }
}
```

i=1 j=1 to n
i=2 j=1 to n
j=3 j=1 to n
i=4 j=1 to n

so for i upto n it will take n2

$T(n) = O(n^2)$

Answer 10 $f(n) = n^k$ $f_2(n) = c^n$

Asymptotic relationship b/w f1 and f2 is Big O

i.e $f_1(n) = O(f(n)) = O(c^n)$

is

$n^k \leq q * c^n$

[q is some constant]